L2: Entry 1 of 1 File: TDBD Oct 1, 1991

TDB-ACC-NO: NN911031

DISCLOSURE TITLE: Performing Hierarchical Authorizations.

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, October 1991, US

VOLUME NUMBER: 34
ISSUE NUMBER: 5
PAGE NUMBER: 31 - 32

PUBLICATION-DATE: October 1, 1991 (19911001)

CROSS REFERENCE: 0018-8689-34-5-31

DISCLOSURE TEXT:

- This article describes a hierarchical scheme based on multiple attributes assigned to users and an algorithm for storing and checking authorizations of a user database on the attributes and the item to which the user is requesting access. - In a system which provides access to sensitive information, there is a need to filter requests based on attributes assigned to the requesting user. Among these attributes are the location of the user, i.e., system, the identification of the user, i.e., userid, and membership within a certain group, i.e., work group . The items to which users request access are referred to as documents although they may include any kind of data storable on a computer. Each document is of a particular document type . Documents are stored in various states, referred to as document status . - Authorization is based on the values of the above-described attributes, i.e., system, work group, user type, userid, document type, and document status . To assign authorities, these attributes are searched as follows: user-type authorities, work group authorities, and any other individual authorities. Thus, for each user, there is a list of document types and, for each type, each possible status. Therefore, for each pair of document type and document status, the user may have authority, for example, to browse, to print, to add, to modify, or to delete documents of the document type and status. (Other user operations are possible.) An algorithm using the IBM Restructured Extended Executor (REXX) language controls the storage and retrieval of the authorization information. The basic information about authorizations is stored in a tabular relational database management system. The storage and manipulation of the authorizations within the computer memory is the principal focus of this description, however. - For each possible document type - document status pair, there is an entry in the DocTypes REXX stem, the entry being structured as DocTypes.DocIndex = DocType + 'they' + DocStatus where they is a delimiter and + is the string concatenation operation (usually shown in REXX as two parallel vertical lines). - The index, DocIndex, of the array is the key to the authorizations that will be stored elsewhere so the DocTypes stem is built at initialization time and filled with all the possible document type - document status pairs. - Similar to the DocTypes stem is a Systems stem that stores userid and system information for each user of the system as Systems.StemIndex = userid + 'they' + System. In this case, the SystemIndex is used as the index into the authorization stem. The authorization stems contain the intersection of the other two stems such that Browse.SystemIndex = 'they' + DocIndex + 'Ù' for each of the document type - document status pairs for which the user is authorized. The five authorization stems, i.e., Browse, Print, Add, Modify and Delete, now indicate whether a user is authorized to perform the associated action. - When a request is received by the program that is

processing the authorizations, the document type and document status of the document to which the user is seeking access is found in the DocTypes stem to get DocIndex. Next, the userid and system of the requesting user are found in the Systems stem to get the system index. Finally, the stem associated with the action to be performed is checked to make sure, e.g., that index(Browse.SystemIndex = 'they' + DocIndex + 'Ù') is not equal to zero.